Tomáš Meravý Murárik
ais:127232

Zenová záhrada

Zadanie č. 1

Algoritmus a jeho princípy

Tento projekt využíva evolučný algoritmus na optimalizáciu pohybu "mnícha" v 2D mape. Algoritmus funguje na princípe genetických algoritmov, kde sa mnísi zlepšujú cez postupné generácie kombináciou tých najlepších z nich a následnou mutáciou. Cieľom je dosiahnuť čo najvyššie skóre tým, že mních prejde čo najväčšiu časť poľa.

Proces evolúcie

1. Populácia) Na začiatku sa vygeneruje populácia ktorej počet závisí od používateľa . Každý jedinec so svojími valstnými pozíciami štartu a krokmi pohybu. V každej generácii sa vyhodnotí skóre všetkých jedincov, ktoré závisí od toho, koľko krokov urobia na mape /koľko miesta zaberú.

2. Breeding) Z každej generácie sa vyberie skupina najlepších jedincov na základe dosiahnutého počtu bodov. Títo jedinci sa následne "množšia"/ ”krížia” - kombinujú sa ich gény (pozície štartov a kroky pohybu) na vytvorenie nových jedincov do ďalšej generácie.

3. Mutácia)Aby sa zabránilo stagnácii evolúcie, do nových jedincov sa pridáva mutácia. Počas mutácie sa náhodne zmení niekoľko génov, čo umožňuje, aby sa vygenerovali nové a lepšie riešenia, ktoré neboli prítomné v predchádzajúcej populácii.

Vlastnosti génov

Každý mních má jeden typ génov:

Tomáš Meravý Murárik
ais:127232

- start_positions :Určujú, na ktorých miestach mních začne svoju cestu. Tieto pozície sú vybrané z okrajov mapy.

Tento systém zaručuje, že každý mních má jedinečnú trasu a rozhodovanie, ale zdieľa podobné vlastnosti s ostatnými v populácii.

Rozhodovanie počas pohybu:

1. Mních sa najprv snaží ísť v smere definovanom génom

-pri začatí pohybu jedinca jeho start_position definuje aj jeho pohyb

```
elif positions[0] == 0:
    self.direction = "d"
    self.prev_move = "d"
elif positions[0] == mapy_shape[0] - 1:
    self.direction = "u"
    self.prev_move = "u".
```

(ak je napr Y jeho pozície rovný 0 tak sa očakáva že mních pôjde zhora smerom dole teda jeho smer je daný ako “d” teda “down”)

2. Ak narazí na prekážku alebo okraj, pokúsi sa hýbať v dostupných smeroch ak mu to pole dovoluje.

```
if not flag:
    if self.direction in ["l", "r"]:
        up = self.can_move([self.position[0] - 1, self.position[1]])
        down = self.can_move([self.position[0] + 1, self.position[1]])
        if up and down:
            self.direction = random.choice(["u", "d"])
        elif not up and not down:
            self.score -= 10
            self.position = []
```

Tomáš Meravý Murárik
ais:127232

```
            return True
        elif up:
            self.direction = "u"
        else:
            self.direction = "d"


    if self.direction == "":
        self.direction = self.steps[self.i]
    self.direction = self.steps[self.i]
    self.i += 1
    if self.i == len(self.steps):
        self.i = 0
```

Teda na začiatku pohybu sa nastaví flag na False ktorý symbolizuje či sa spravil pohyb . Ak je na konci pohybu tento flag False tak to znamená že pohyb nebol urobený. V takomto prípade sa zmení direction a mních sa pokúsi ďaľej hýbať.

3. Ak sa nemôže pohnúť, získava penalizáciu.

Ako je ukázané vyššie v kóde ,ak ani po vyskúšaní všetkých pozícií neurobí pohyb tak sa predpokladá že je zaseknutý na mieste a preto sa mu odpočíta 10 bodov z jeho skóre a odíde z move funkcie.

Tvorba novej generácie

Po každej generácii sú vybraní jedinci s najvyšším skóre. Títo jedinci sa krížia, pričom kombinujú svoje gény na tvorbu nových jedincov. Tento proces zahŕňa:

- **Kríženie génov**: Dvaja najlepší mníchovia si vymenia časť svojich génov, aby vytvorili potomstvo.

```
    x = random.randint(0, number_of_positions - 1)
    # skombinujeme pozície dvoch jedincov
    newpositions = (
        top_kittens[i][1][0:x]
```

Tomáš Meravý Murárik
ais:127232

```
            + top_kittens[j][1][x : number_of_positions - 1]
        )
```

- **Mutácie**: Do nových potomkov sa náhodne pridáva mutácia, ktorá mení niektoré kroky alebo pozície štartov.

```
    for i in range(mutation_steps):
        position = random.randint(0, number_of_steps - 1)
        self.steps[position] = random.choice(["l", "u", "r", "d"])
```

-Mutácie sa dajú vyberať dvomi spôsobmi

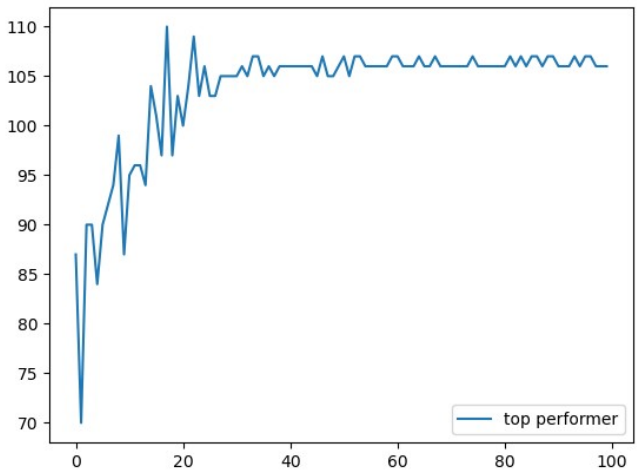
1) kompletným anihilovaním predošlej generácie

```
for i in range(len(top_kittens)):
    for j in range(len(top_kittens)):
```

spôsob ktorý je implementovaný teraz prejde cez všetkých jedincov

```
        newpositions = (
            top_kittens[i][1][0:x]
            + top_kittens[j][1][x : number_of_positions - 1]
        )
```

a následnej ich všetkých spáruje pomocou ich génou.

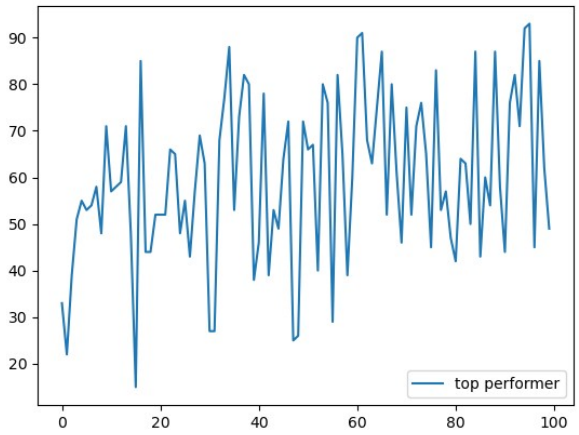


2) ruleta

Tomáš Meravý Murárik
ais:127232

```
if pick == 2:
    for i in reversed(range(number_of_kittens)):
        ([dummy_array.append(i) for _ in range(number_of_kittens - i)])
    print(dummy_array)
```

Ak je pick premenná 2 tak sa aplikuje výber pomocou rulety kde najlepší jedinec má najväčšiu šancu na výber a najhorší jedinec má najmenšiu šancu na výber



Ako je viditelné tak algoritmus aj keď pomaľšie a viac nekonzistentne sa stále zlepšuje.

Parametre a ich nastavenie

- number_of_positions: Určuje, koľko rôznych štartovacích pozícií má každý mních.

- number_of_generations: Určuje, koľkokrát sa vykoná evolučný proces.

- kitten_count: Počet jedincov v jednej generácii.

- mutácie: Určuje, koľko génov sa v každej generácii zmutuje.

-pick : Pri 1 sa použije výber podľa najlepších jedincov a pri 2 ruleta

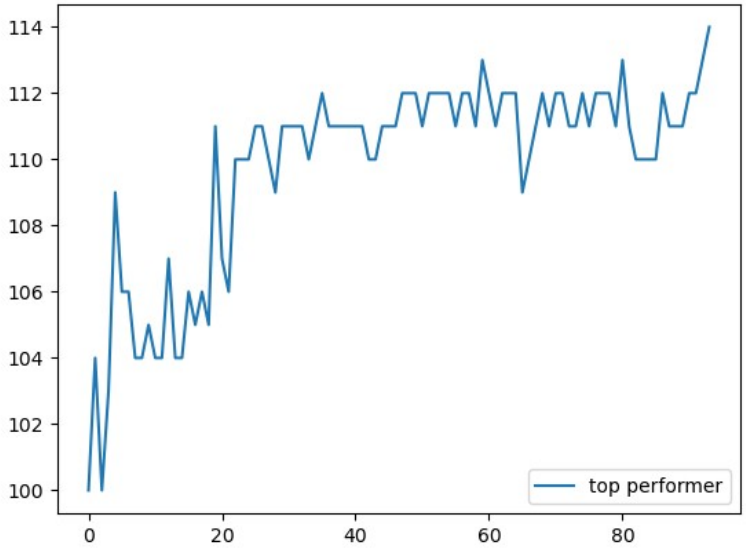
Tomáš Meravý Murárik
ais:127232

- Rocks:Určuje kde sa nachádzajú kamene

- mapy_shape: Určuje veľkosť Y a X veľkosť mapy

Hodnotenie a výsledky

Systém evolúcie vedie k tomu, že mnísi sa v každej generácii zlepšujú. Ako generácie pokračujú, najlepší jedinci postupne dosahujú lepšie skóre. Najvyššie dosiahnuté skóre je 114 a aj keď to môj program dosahuje nie tak často ako by som si prial , dokáže ho nájsť (ako bolo ukázané na cviku).



(2 génová mutácia pozície a 2 génová mutácia krokov)

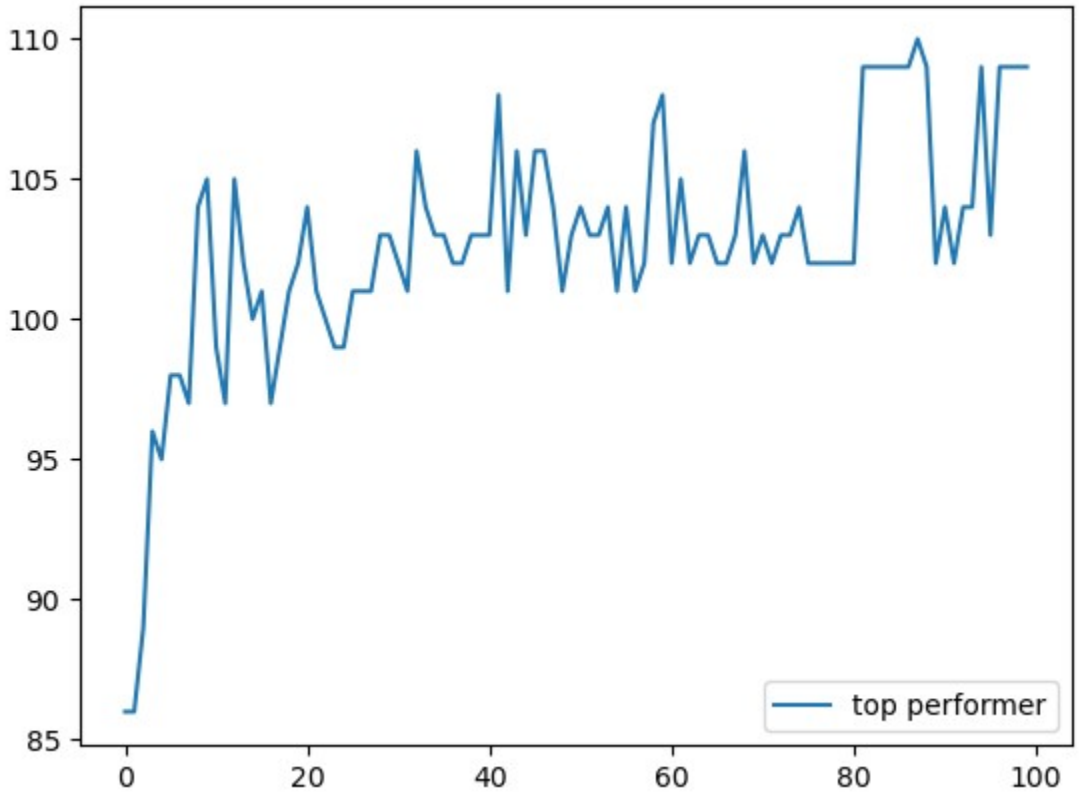
Výsledky pre rôzne parametre

- Pri vyššom počte generácií sa zvyšuje pravdepodobnosť, že mnísi dosiahnu vysoké skóre, pretože majú viac času na evolúciu.

- Väčší počet jedincov v populácii umožňuje viac kombinácií a lepšie riešenia, ale zároveň drasticky zvyšuje časovú náročnosť simulácie.

- Zvýšená mutácia môže pomôcť objaviť nové riešenia, ale ak je mutácia príliš veľká, môže to viesť k náhodným a menej efektívnym riešeniam a z mojích skúseností ku veľmi nekonzistentným výsledkom.

Tomáš Meravý Murárik
ais:127232

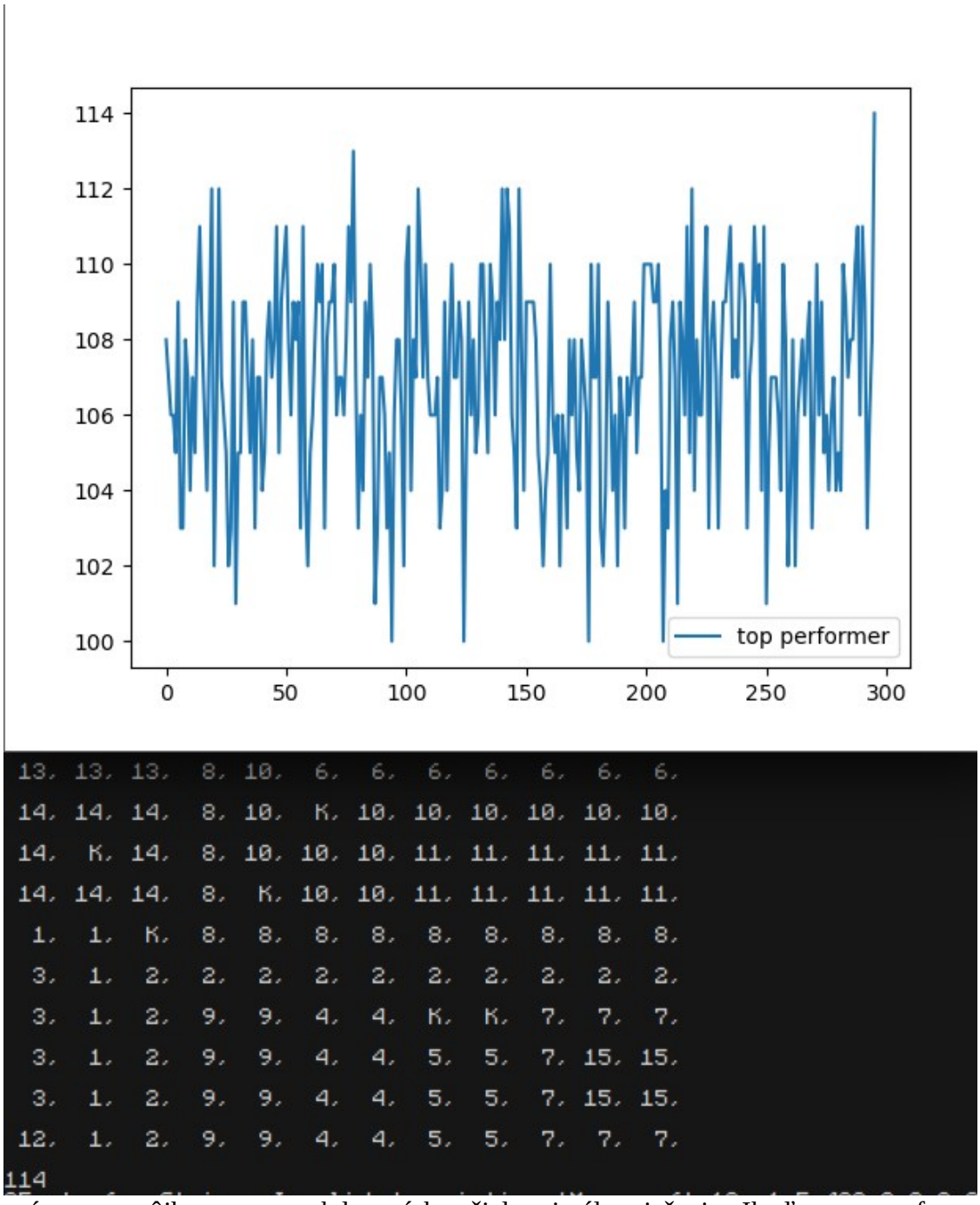


(pre porovnanie : tento príklad je pri 100 generáciách , 2 génovej mutácií a 8 génovej krokovej mutácií)

Zlepšovanie a dolaďovanie

Systém sa dá ďalej vylepšiť zmenou parametrov, ako sú počet mutácií, počet krokov, alebo spôsob, akým sa kombinujú gény. Pri optimalizácii sa ukázalo, že je dobré udržiavať mutácie na nízkej úrovni, aby nedošlo k príliš veľkej variabilite medzi generáciami. Taktiež sa dá optimalizovať výber smeru pri narazení na prekážku nahradením listu smerov (steps) napr listom boolean hodnôt .Posledné vylepšenie by mohlo byť fixnutie chyby kde pri náhodných spusteniach mních neodíde z poľa ale namiesto toho si sadne na kameň a považuje to za valídny pohyb.

Tomáš Meravý Murárik
ais:127232



(Ukážka výstupu z môjho programu kde mních našiel optimálne riešenie . Ikeď sa to z grafu nezdá lebo prvá generácia dosiahla impresívne výsledky tak stále sa cez celých 300 generácií jedinec učil a menil)